Semantic Web Project

Muhammad Hassaan Anwar

190754334

Request:

The Pellet Reasoner is quite buggy sometimes, I want to request that, when checking this coursework, if the reasoner is not giving desired outputs, please try stopping the reasoner and starting it again.

.

SRSCP: Semantic Reasoner for Smart Car Purchases

Muhammad Hassaan Anwar
190754334
Queen Mary University of London:
School of Electronic Engineering and Computer Science

I. Introduction

There are various online platforms that provide services in car purchases and assists us in making choices regarding buying a new car. However, most of them are biased as they are getting sponsored by a specific car's brand and therefore does not make a fair comparison between several models of cars. SRSCP is an initiative with an aim to provide a bias free semantic reasoner that is able to make fair comparison between cars stored in its knowledge base. To achieve this, I have divided the task into several steps, knowledge acquisition is the step in which I generate unorganized knowledge in the form of terms e.g (instances, verbs). Which is passed to the second stage, the knowledge conceptualization phase then arranges these terms into a structured model that describes the corresponding Domain.

All the concepts and their associated relations, attributes, and instances from the knowledge conceptualization step are codified into a formal language in the knowledge formalization step. OWL, the standard language for ontology construction, is used to formalize the knowledge. This is supported by Protégé 5.5.0, an open-source software for building intelligent systems, based on OWL language
SWRL is embedded to enable expressivity (i.e., rules and logic) in OWL.

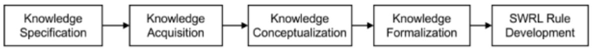


Figure 1 The development process of SRSCP

II. Project Aims

Presented here is a summary of some of the aims we hope to achieve undertaking this project:

- Enabling Logical use of concepts and sub-concepts to furnish the ontology.
- Correct and effective use of object properties (including constraints and characteristics such as **functional, transitive and irreflexive**)
- Extensive use of SWRL rules (use of object, data properties and SWRL built-ins)
- Formulating a correct A-Box with a sufficient number of individuals to use with the defined logic rules (expecting majority of the relations defined by the data properties.

III. Methodology

The Data is collected from google using several online websites and may not be an correspond to a car's true specification in the real world. Individuals that has been extracted from online websites are:

Table I Data

Cars Names	Company Names
Audi A8	**Audi**
Honda Odyssey	**Honda**
Range Rover	**Land Rover**
Mercedes AMG A 45	**Mercedes**
Mercedes AMG A 45-Hybrid	**Mercedes**
Mercedes C Class	**Mercedes**
Mercedes AMG A 43	**Mercedes**
Porsche Taycan	**Porsche**
Porsche 718	**Porsche**

So in total our knowledge base has 1 car of Audi Company, 1 car of Honda Company, 1 car of Land Rover company, 4 cars of Mercedes, and 2 cars of Porsche.

The SRSCP represents the concepts, relations, attributes, and instances for performing Semantic Reasoning. The SRSCP, supported by OWL axioms and SWRL rules, becomes the foundation for automated SR provided the knowledge base. The visualization of SRSCP is shown in Fig. 2. There are Twelve main classes in the SRSCP model.

The information was collected with respect to the following properties of the cars.

Table II Object Properties

Object Properties	Domain	Range
Has Ancestor	Cars	Cars
Body Of	Cars	Body
Cheaper Than	Cars	Cars
Fuel Type Of	Fuel	Cars
Has Wheel Drive	Cars	Wheel Drive
Has Rival	Cars	Cars
Manufactured By	Cars	Company
Has Seats	Cars	Has Seats
Has Transmission	Cars	Transmission
Has Hybrid Version	Cars	Cars
Powerful Than	Cars	Cars
Expensive Than	Cars	Cars

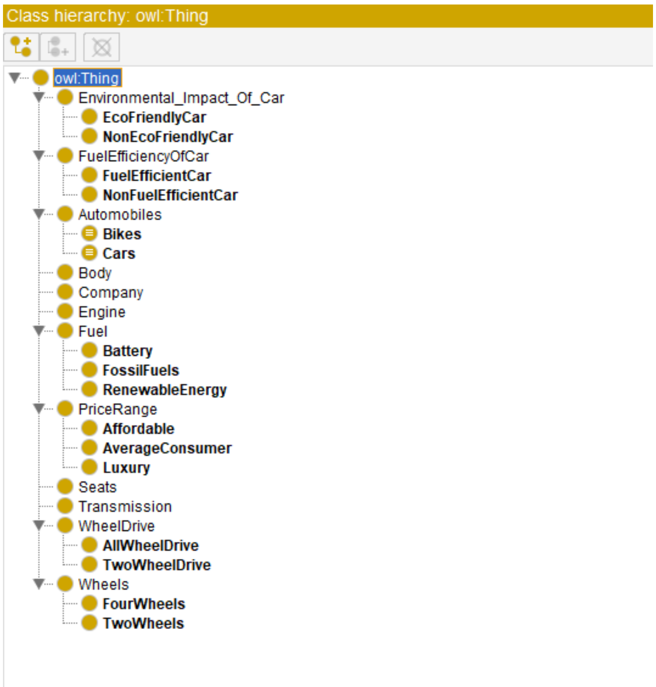
The Data Property for each car added were:

Table III Data Properties

Data Property	Domain	Range
TopSpeed	Cars	Integer
Weight	Cars	Integer
Boot	Cars	Boolean
CO2Emissions	Fuel	Integer
HoursePower	Cars	Integer
Mileage	Cars	Integer
Price	Cars	Integer
SecondFuel	Cars	Literal
Manufacturing Year	Cars	Integer
Model Code	Cars	Integer

IV. Implementation

To construct the T-Boxes, I have used the help of SWRL rules, that will help classify cars on the basis of their environment-friendliness, Fuel-Efficiency, and Affordability.
Moreover, Effective use of Transitive, asymmetric, irreflexive and symmetric rules have been put in practice for better reasoning capabilities.



Class hierarchy: owl:Thing

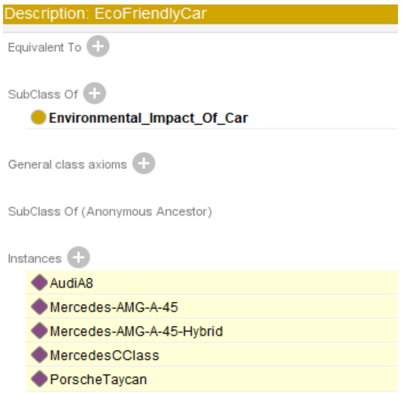
- owl:Thing
  - Environmental_Impact_Of_Car
    - EcoFriendlyCar
    - NonEcoFriendlyCar
  - FuelEfficiencyOfCar
    - FuelEfficientCar
    - NonFuelEfficientCar
  - Automobiles
    - Bikes
    - Cars
  - Body
  - Company
  - Engine
  - Fuel
    - Battery
    - FossilFuels
    - RenewableEnergy
  - PriceRange
    - Affordable
    - AverageConsumer
    - Luxury
  - Seats
  - Transmission
  - WheelDrive
    - AllWheelDrive
    - TwoWheelDrive
  - Wheels
    - FourWheels
    - TwoWheels

Figure 2

A. *Eco Friendly Car:*

To define an Eco Friendly Car, the SWRL rule is as follows:

Cars(?p) ^ CO2Emissions(?p, ?x) ^ swrlb:lessThan(?x, 150) -> EcoFriendlyCar(?p)



Description: EcoFriendlyCar

Equivalent To

SubClass Of
- Environmental_Impact_Of_Car

General class axioms

SubClass Of (Anonymous Ancestor)

Instances
- AudiA8
- Mercedes-AMG-A-45
- Mercedes-AMG-A-45-Hybrid
- MercedesCClass
- PorscheTaycan

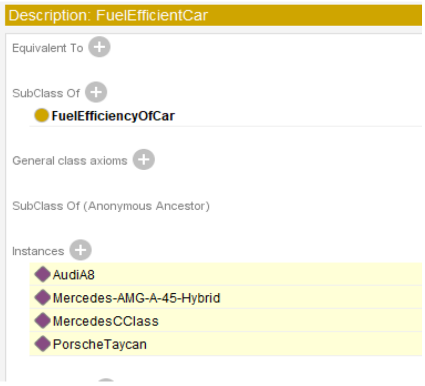
Figure 3

Which means that Cars whose CO2Emissions are lesser 150 gm/km will be categorized as Eco Friendly Cars

B. *Fuel Efficient Car:*

To define a fuel-efficient car, the SWRL rule is as follows:

Cars(?q) ^ Mileage(?q, ?x) ^ swrlb:greaterThan(?x, 40) -> FuelEfficientCar(?q)



Description: FuelEfficientCar

Equivalent To

SubClass Of
- FuelEfficiencyOfCar

General class axioms

SubClass Of (Anonymous Ancestor)

Instances
- AudiA8
- Mercedes-AMG-A-45-Hybrid
- MercedesCClass
- PorscheTaycan

Figure 4

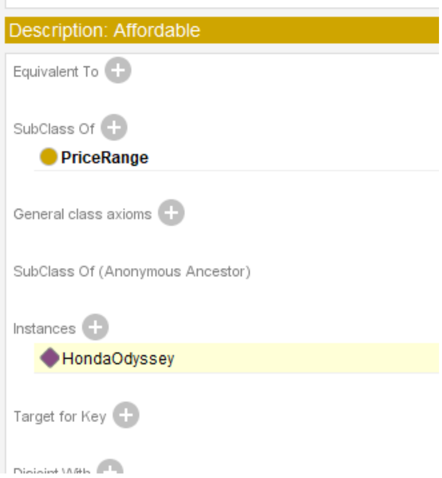
Which means that car whose Mileage is greater than 40 Miles, will be categorized as Fuel Efficient.

C. *Affordable Car:*

To define an affordable car, the SWRL rule is as follows:

Cars(?p) ^ price(?p, ?x) ^ swrlb:lessThan(?x, 10000) -> Affordable(?p)

Which means that cars whose price is less than 10000 British Pounds are categorized as affordable.



Description: Affordable

Equivalent To

SubClass Of

PriceRange

General class axioms

SubClass Of (Anonymous Ancestor)

Instances

HondaOdyssey

Target for Key

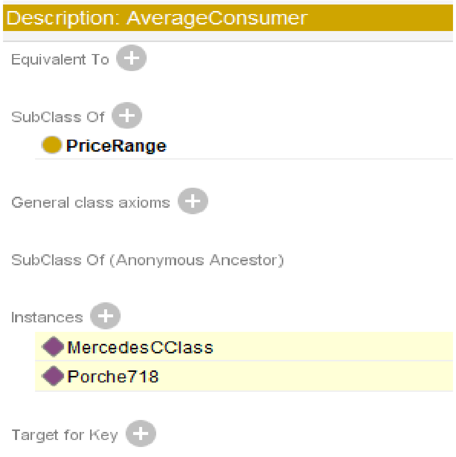
Disjoint With

Figure 5

D. *Average Consumer Car:*

To define an Average Consumer Car, the SWRL rule is as follows:

Cars(?p) ^ price(?p, ?x) ^ swrlb:lessThan(?x, 50000) ^ swrlb:greaterThan(?x, 10000) -> AverageConsumer(?p)



Description: AverageConsumer

Equivalent To

SubClass Of

PriceRange

General class axioms

SubClass Of (Anonymous Ancestor)

Instances

MercedesCClass

Porche718

Target for Key

Figure 6

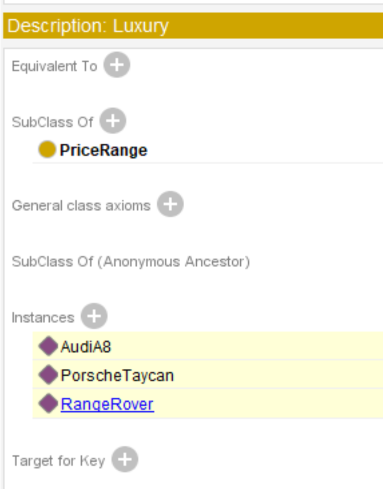
Which means that cars with price greater than 10000 but lesser than 50000 GBP will be categorized as Average Consumer Car.

E. *Luxury Car:*

To define a Luxury Car, the SWRL rule is as follows:

Cars(?p) ^ price(?p, ?x) ^ swrlb:greaterThan(?x, 50000) -> Luxury(?p)

Which means that cars with price greater than 50000 GBP will be categorized as Luxury Car.



Description: Luxury

Equivalent To

SubClass Of

PriceRange

General class axioms

SubClass Of (Anonymous Ancestor)

Instances

AudiA8

PorscheTaycan

RangeRover

Target for Key

Figure 6

F. *Powerful than* - ***Transitive property***

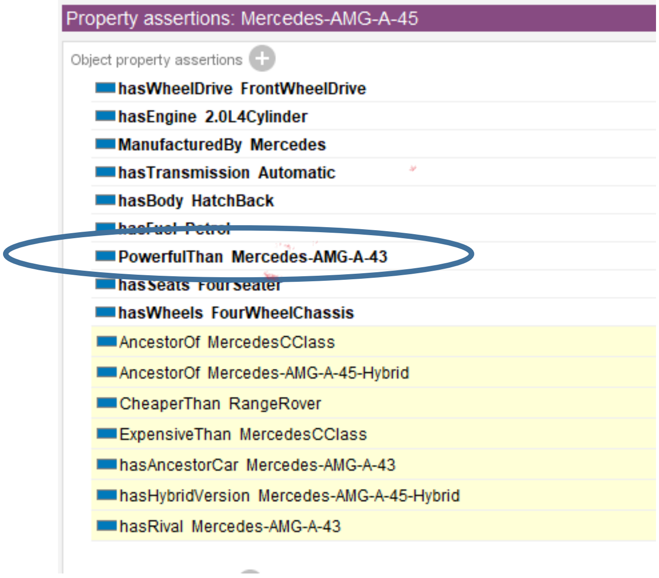
I have enabled transitive ruling in my ontology. For instance, car A has engine X and car B has engine Y, and car C has Engine Z. Engine X is more powerful than Engine Y and Engine Y is more powerful than Engine Z.

Then it is evident that Car A is more powerful than Car C since it has a more powerful engine then Car B.

To explain the transitive property, I have taken three cars from our knowledge base:

- Mercedes AMG A 43
- Mercedes AMG A 45
- Mercedes C Class

In figure below we defined that Mercedes AMG A 45 is Powerful than Mercedes AMG A 43.



Property assertions: Mercedes-AMG-A-45

Object property assertions

hasWheelDrive FrontWheelDrive

hasEngine 2.0L4Cylinder

ManufacturedBy Mercedes

hasTransmission Automatic

hasBody HatchBack

hasFuel Petrol

PowerfulThan Mercedes-AMG-A-43

hasSeats FourSeater

hasWheels FourWheelChassis

AncestorOf MercedesCClass

AncestorOf Mercedes-AMG-A-45-Hybrid

CheaperThan RangeRover

ExpensiveThan MercedesCClass

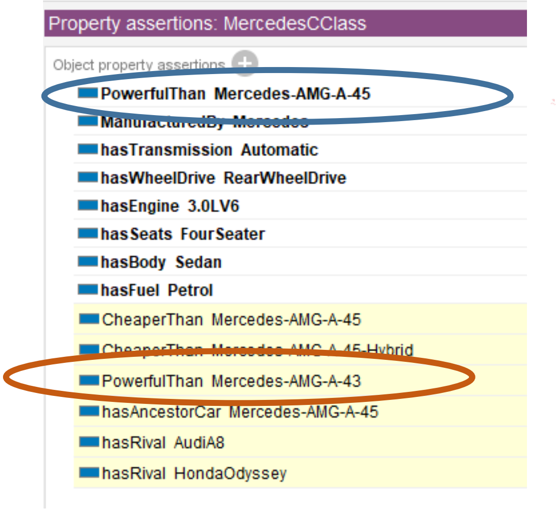
hasAncestorCar Mercedes-AMG-A-43

hasHybridVersion Mercedes-AMG-A-45-Hybrid

hasRival Mercedes-AMG-A-43

Figure 7

In figure 8 we define, Mercedes C Class to be more powerful than Mercedes AMG A 45 version, hence the reasoner will reason that Mercedes C Class should be more powerful than Mercedes AMG A 43 too because of the transitive rule.



Property assertions: MercedesCClass

Object property assertions

PowerfulThan Mercedes-AMG-A-45

ManufacturedBy Mercedes

hasTransmission Automatic

hasWheelDrive RearWheelDrive

hasEngine 3.0LV6

hasSeats FourSeater

hasBody Sedan

hasFuel Petrol

CheaperThan Mercedes-AMG-A-45

CheaperThan Mercedes-AMG-A-45-Hybrid

PowerfulThan Mercedes-AMG-A-43

hasAncestorCar Mercedes-AMG-A-45

hasRival AudiA8

hasRival HondaOdyssey

Figure 8

G. *Manufactured By: **Functional***

The object property Manufactured By, has been declared functional, as a car can just be manufactured by only one company, therefore calling this property as Functional is really meaningful..

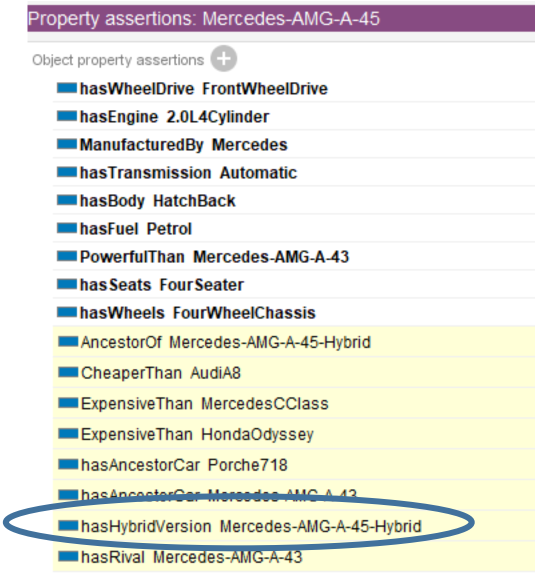
H. *Has Hybrid Version – **Irreflexive Property***

To determine if a car is a Hybrid version of another car then our swrl rule will look like the following:

hasFuel(?x, ?y) ^ Model(?x, ?c) ^ ManufacturedBy(?x, ?z) ^ManufacturerOf(?z, ?o) ^ differentFrom(?o, ?x) ^ SecondFuel(?o,?p)^swrlb:contains(?p,"Battery")^Model(?o,?cc)^swrlb:equal(?c,?cc) -> hasHybridVersion(?x, ?o)

Which means that if a car X is produced by a company Y and the Car X is powered by some fuel and has a model name, If the Company Y makes another Car with the name Z, and if that car has the same fuel along with a Second Fuel which has name “Battery” and the model-variant number of both cars is the same. This means that the second car is the Hybrid version of first car.

Note: If the assertion doesnot appear while running the reasoner, please stop the reasoner and restart it again.



Property assertions: Mercedes-AMG-A-45

Object property assertions

hasWheelDrive FrontWheelDrive

hasEngine 2.0L4Cylinder

ManufacturedBy Mercedes

hasTransmission Automatic

hasBody HatchBack

hasFuel Petrol

PowerfulThan Mercedes-AMG-A-43

hasSeats FourSeater

hasWheels FourWheelChassis

AncestorOf Mercedes-AMG-A-45-Hybrid

CheaperThan AudiA8

ExpensiveThan MercedesCClass

ExpensiveThan HondaOdyssey

hasAncestorCar Porche718

hasAncestorCar Mercedes-AMG-A-43

hasHybridVersion Mercedes-AMG-A-45-Hybrid

hasRival Mercedes-AMG-A-43

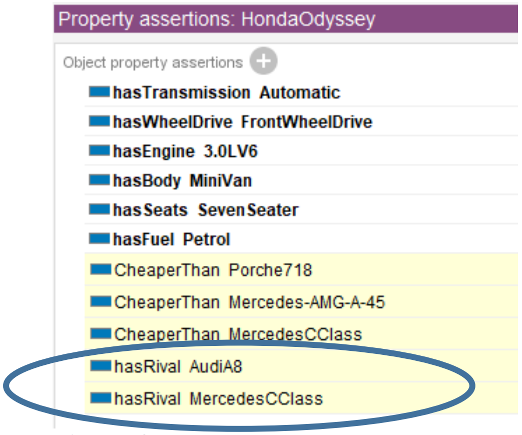
Figure 9

I. *Has Rival Car – **Symmetric Property:***

To define a Rival car, I have assumed the definition that Rival cars are cars that are different from each other but share the same engine, for example Audi A8 and Mercedes C Class are the “Rival Cars” according to our knowledge base. I have used the following SWRL rule:

hasEngine(?x, ?y) ^ EngineOf(?y, ?z) ^ differentFrom(?x, ?z) -> hasRival(?x, ?z)

Which states that if a car has engine Y and another car in the knowledge base has the same engine Y then both cars are Rivals or “ close competitors “ of each other.



Property assertions: HondaOdyssey

Object property assertions

hasTransmission Automatic

hasWheelDrive FrontWheelDrive

hasEngine 3.0LV6

hasBody MiniVan

hasSeats SevenSeater

hasFuel Petrol

CheaperThan Porche718

CheaperThan Mercedes-AMG-A-45

CheaperThan MercedesCClass

hasRival AudiA8

hasRival MercedesCClass

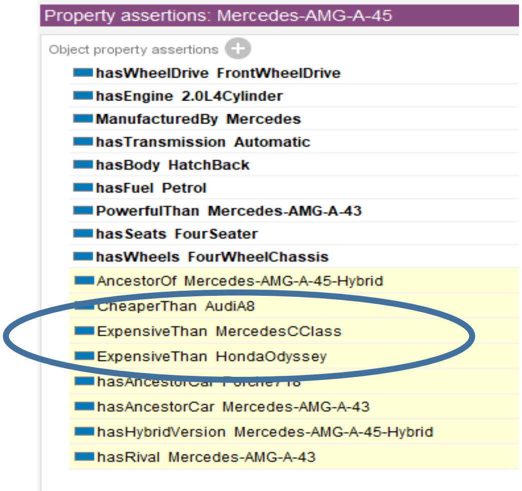
Figure 10

J. *Expensive than – Irreflexive*

To determine whether a car X is expensive than Y, I have used a SWRL rule:

price(?p, ?x) ^ price(?q, ?y) ^ swrlb:greaterThan(?x, ?y) ^ differentFrom(?p, ?q) -> ExpensiveThan(?p, ?q)

Which means that if a car P has some price and if another car Q has a greater price than P then Q is expensive than P.



Property assertions: Mercedes-AMG-A-45

Object property assertions ⊕

hasWheelDrive FrontWheelDrive
hasEngine 2.0L4Cylinder
ManufacturedBy Mercedes
hasTransmission Automatic
hasBody HatchBack
hasFuel Petrol
PowerfulThan Mercedes-AMG-A-43
hasSeats FourSeater
hasWheels FourWheelChassis
AncestorOf Mercedes-AMG-A-45-Hybrid
CheaperThan AudiA8
ExpensiveThan MercedesCClass
ExpensiveThan HondaOdyssey
hasAncestorCar Porche718
hasAncestorCar Mercedes-AMG-A-43
hasHybridVersion Mercedes-AMG-A-45-Hybrid
hasRival Mercedes-AMG-A-43

Figure 11

K. *Cheaper than – InverseOf Expesive Than*

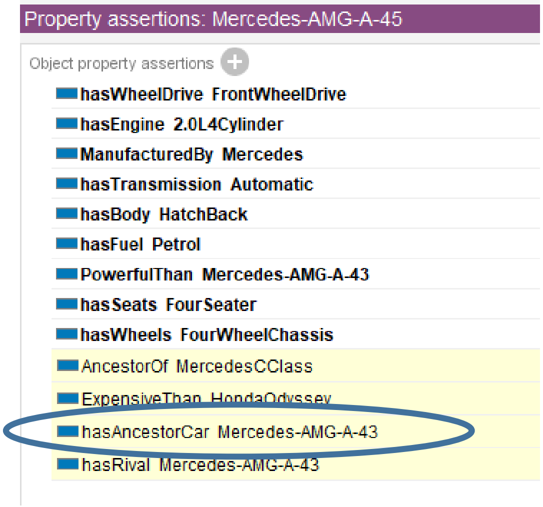
This rule is just an inverse of Expensive than object property, such that if a car X is expensive than car Y, then the car Y is cheaper than car X.

L. *Has Ancestor Car – Asymmetric*

To define an Ancestor Car, I have used the following rule:

ManufacturingYear(?p, ?x) ^ ManufacturedBy(?p, ?y) ^ ManufacturerOf(?y, ?z) ^ ManufacturingYear(?z, ?w) ^ swrlb:greaterThan(?w, ?x) ^ differentFrom(?p, ?z) -> hasAncestorCar(?z, ?p)

Which means that if a car P has a manufacturing year X, and there is another car Z which has a manufacturing year W, the car P and Z are made by the same company Y. The manufacturing Year of Car Z is greater than car P then car P is an ancestor car of car Z.



Property assertions: Mercedes-AMG-A-45

Object property assertions ⊕

hasWheelDrive FrontWheelDrive
hasEngine 2.0L4Cylinder
ManufacturedBy Mercedes
hasTransmission Automatic
hasBody HatchBack
hasFuel Petrol
PowerfulThan Mercedes-AMG-A-43
hasSeats FourSeater
hasWheels FourWheelChassis
AncestorOf MercedesCClass
ExpensiveThan HondaOdyssey
hasAncestorCar Mercedes-AMG-A-43
hasRival Mercedes-AMG-A-43

Figure 12

V. Prospects Of the project

This project has great prospects if implemented on a larger scale. It can reason and assist between different brands within the same price range and help customers buy the best product to match their needs. It can compare features such as Seats, Fuel-Type, Engine, Mileage, hybrid engine compatibility, Fuel Consumption. Allowing us to buy the best car for the budget.

VI. Problems Faced in Making the project

A. *Selecting Ontology Topic:*

The biggest hurdle in making the project was selecting the topic to work with, Initially I started working with an ontology of Aquatic Animals. I had to redesign and restructure my project idea, go through couple of research papers to lay the foundations of my coursework. Initially my topic was “Aquatic Animals”. Later I changed my project topic to “Cars”, I realized it will be a more meaningful product to work on. However, this topic required a lot of research regarding car models and their components which was time consuming for me.

B. *Confusion between Irreflexive, and asymmetric property:*

One of the key things that caused a lot of confusion was use of irreflexive vs asymmetric properties. In almost every documentation available on the internet it is really hard to distinguish between the two. For example “Mother Of” is an irreflexive property, but it can also be called as an asymmetric property. I was not able to find the answer of this question in various forums and even after reading couple of documentations. However, in the end I went through this documentation w3.org/TR/owl2-syntax/.
I found it really helpful to clear my understanding.

C. *Reasoner Issue:*

I was trying to run the Hermit Reasoner in the beginning, and it took me a while to realize that none other than "Pellet" reasoner works and provide support for SWRL rules in Protégé. Other Reasoners were not able to comprehend SWRL rules **(Resolved)**

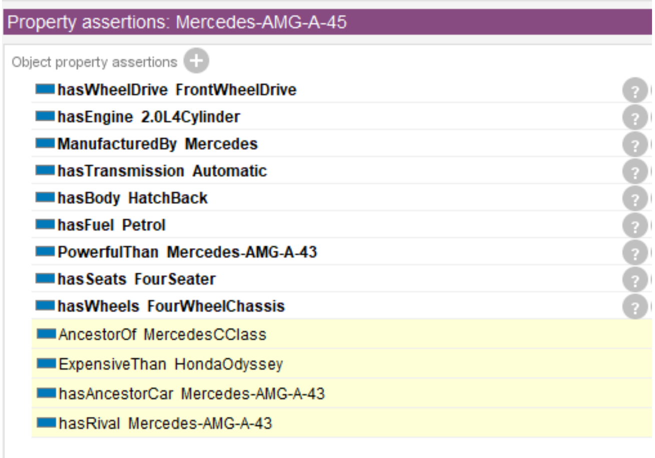
D. *Description Logic Vs SWRL Ruling Confusion:*

I was confused if I needed to write Description Logic definitions for the classes for which I I am already using SWRL rules, and the answer was no. (I contacted Sir Mustafa for this query). **(Resolved)**

E. *Reasoner Showing Inconsistency: (Unresolved)*

The Pellet reasoner gives inconsistent results, sometimes I have to run the reasoner multiple times to get the results. For example: according to my ontology, the *car "Mercedes AMG A 45"* has a hybrid variant. But sometimes the reasoner shows this and sometimes it does not. As shown below:

Attempt 1:



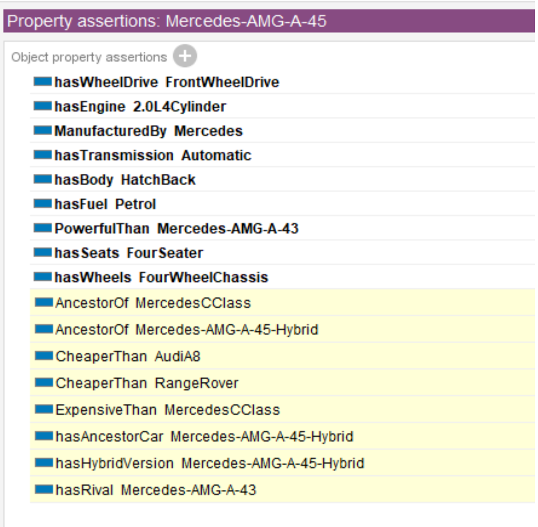
Property assertions: Mercedes-AMG-A-45

Object property assertions

hasWheelDrive FrontWheelDrive
hasEngine 2.0L4Cylinder
ManufacturedBy Mercedes
hasTransmission Automatic
hasBody HatchBack
hasFuel Petrol
PowerfulThan Mercedes-AMG-A-43
hasSeats FourSeater
hasWheels FourWheelChassis
AncestorOf MercedesCClass
ExpensiveThan HondaOdyssey
hasAncestorCar Mercedes-AMG-A-43
hasRival Mercedes-AMG-A-43

As it is evident here that a couple of assertions are missing here.

Attempt 2:
(After stopping reasoner and restarting it)



Property assertions: Mercedes-AMG-A-45

Object property assertions

hasWheelDrive FrontWheelDrive
hasEngine 2.0L4Cylinder
ManufacturedBy Mercedes
hasTransmission Automatic
hasBody HatchBack
hasFuel Petrol
PowerfulThan Mercedes-AMG-A-43
hasSeats FourSeater
hasWheels FourWheelChassis
AncestorOf MercedesCClass
AncestorOf Mercedes-AMG-A-45-Hybrid
CheaperThan AudiA8
CheaperThan RangeRover
ExpensiveThan MercedesCClass
hasAncestorCar Mercedes-AMG-A-45-Hybrid
hasHybridVersion Mercedes-AMG-A-45-Hybrid
hasRival Mercedes-AMG-A-43

After stopping the reasoner and starting it again, it now shows the Hybrid variant and solves the problem. However, I was not able to figure out the reason behind this.

VII. ACKNOWLEDGEMENT:

I am extremely thankful to all the demonstrators, lecturers for giving us their valuable time in explaining concepts and transferring knowledge through this coursework.

X-----X------X-------X -------X-----X------X-------X

End Of Report.